

A hybrid Lagrangean metaheuristic for the two-machine cross-docking flow shop scheduling problem

Gabriela B. Fonseca^a, Thiago H. Nogueira^b, Martín Gómez Ravetti^{a,*}

^a*Department of Production Engineering, Universidade Federal de Minas Gerais, Av. Antônio Carlos, 6627, CEP 31270-901, Belo Horizonte, MG, Brazil.*

^b*Department of Production Engineering, Universidade Federal de Viçosa, Rodovia MG-230, CEP 38810-000, Rio Paranaíba, MG, Brazil.*

Abstract

Cross-docking is a logistics strategy that minimizes the storage and picking functions of conventional warehouses. The objective is to unload the cargo from inbound trucks and directly load it into outbound trucks, with little or no storage. The success of the strategy depends on an efficient transshipment operation. This work undertakes a study of truck scheduling in a cross-docking center. The problem is modeled as a two-machine flow shop scheduling problem with precedence constraints, with the objective of minimizing the makespan. The proposed method is based on a Lagrangean relaxation solved by a Volume algorithm over a time-indexed formulation. We use polynomial time heuristics for generating efficient upper and lower bounds in a computationally efficient time, outperforming current results in the literature for small and large size instances.

Keywords: Logistics, Truck scheduling, Cross-docking, Lagrangean Relaxation.

*Corresponding author.

Email addresses: gabrielabragafonseca@gmail.com (Gabriela B. Fonseca), thiagoh.nogueira@ufv.br (Thiago H. Nogueira), martin.ravetti@dep.ufmg.br (Martín Gómez Ravetti)

1. Introduction

The current market environment, characterized by an increasingly fierce competition, the globalization of the economy and an accelerated technological revolution has led companies to improve their production, logistics and distribution systems.

More efficient logistics systems are required, customers are demanding better services, because of the increasing necessity for direct-to-customers deliveries, but the cities exponential growth also requires green logistics systems, avoiding traffic congestions and other types of environmental pollution [49].

A Cross-docking Distribution Center (CDC) is a logistics technique widespread throughout the world. Several well-known companies such as retail chains (Walmart [51]), mailing companies (UPS [27]), automobile manufacturers (Toyota [56]) and logistics providers ([29], [30]) have gained considerable competitive advantages by using CDC. The main idea behind cross-docking centers is to receive products from different suppliers or manufacturers and consolidate them to common final delivery destinations. In comparison to traditional warehouses, a CDC is managed with minimal handling and with little or no storage between unloading and loading of goods. This practice can serve different goals: the consolidation of shipments, a shorter delivery lead time, the reduction of costs, etc. Readers are referred to [32], for a survey discussing industry practices and CDC problem characterization.

A schematic representation of processes at a CDC, is illustrated in Figure 1. Firstly, incoming trucks arrive at the yard of the CDC, if the number of trucks is higher than the number of docks, some of them have to wait in a queue until further assignment. Secondly, after being docked, goods of the inbound trucks are unloaded, scanned, sorted, moved across the dock and loaded into outbound trucks for an immediate delivery elsewhere in the distribution chain. Once an outbound (inbound) truck is completely loaded (unloaded), the truck is removed from the dock, replaced by another truck and the course of action repeats.

In a conventional distribution center, five operations are usually carried out

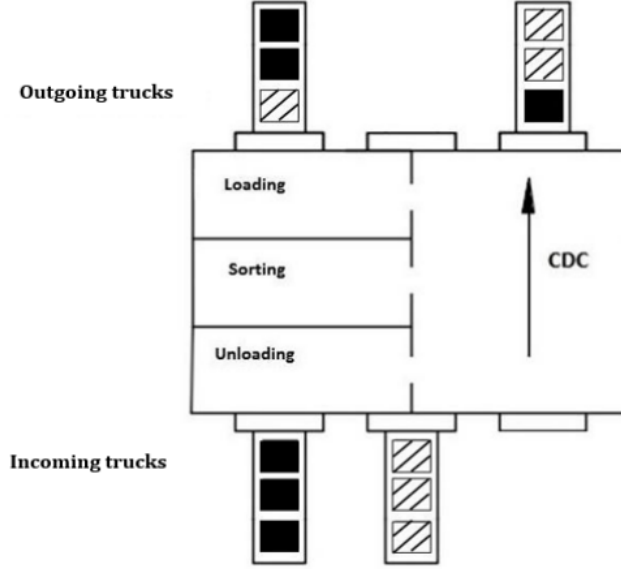


Figure 1: Schematic representation of a CDC.

when managing products: receiving, sorting, storing, picking and shipping operations. All of which can be done consecutively or in a particular order. As a result of applying a cross-docking system, the cost of both storage and products picking can be considerably reduced by consolidating the inbound and outbound trucks flows. The benefits of cross-docking centers are clearly perceived: reduced costs (warehousing, inventory-holding, handling, and labour costs), shorter delivery lead times (from supplier to customer, increase of throughput), improved customer service and customer satisfaction, reduced of storage space, faster inventory turnover, fewer overstocks, reduced risk of loss and damage, etc. However, efficient transshipment processes and careful operations planning become indispensable within a CDC, where inbound and outbound flows need to be synchronized to keep the terminal storage as low as possible and on-time deliveries are ensured. Many articles in the literature develop computerized scheduling procedures, which have achieved good results (See examples in Table 1).

In this work, we consider a time-indexed integer programming formulation

for a flow shop scheduling problem with cross-docking precedence constraints, in a center with two docks. The problem denoted as $F2|CD|C_{max}$ is considered strongly NP-hard (See [20]). Computational experiments show that the model is efficient when solving small problems.

A Hybrid Lagrangean Metaheuristic approach is also proposed and tested. The method can efficiently generate strong lower and upper bounds. Results are compared to the best heuristic introduced by Chen and Lee [20], which is based on Johnson’s algorithm. Better solutions are consistently obtained for all instances tested. The article is organized as follows, an overview of related works is shown in Section 2. The mathematical model and the Hybrid Lagrangean Metaheuristic Framework are described in Section 3, and 4, respectively. Computational experiments are reported in Section 5. Finally, discussions and conclusions are drawn in Section 6.

2. Literature Review

The increasing use of cross-docking techniques has motivated several authors to investigate new ways to improve the design and tactical operations of CDCs. In the literature, several decision problems are studied, some of them concerning strategical and tactical decisions, and others with operational decisions.

Boysen and Fliedner [16], and Belle et al. [12] focus on the review of the existing literature of cross docking problems. According to [15], decision problems in a cross-docking center can be allocated according to the following classification, ordered from strategic to operational levels: location of cross-docking centers, layout, vehicle routing, dock’s assignments of destinations, truck scheduling and resource scheduling inside the center. Here we focus on truck scheduling problems, where the aim is deciding where and when should the trucks be processed.

Examples of strategic decisions in a CDC can be found in many works. Location of a cross-docking is analyzed in Campbell [19], Klose and Drexl [31], Chen et al. [23], and Chen [22]. The layout of cross docking centers is studied in Gue [29], Bartholdi and Gue [9] and Vis and Roodbergen [34].

Some other works deal with the vehicle routing as Oh et al. [44] and Wen et al. [24]. Regarding operational decisions, some works consider the dock assignment problem: Belle et al. [12], Boysen and Fliedner [16], Tsui and Chang [53] [54], Bozer and Carlo [18] and Gue [29].

In a CDC, scheduling decisions are particularly important to ensure a rapid turnover and on-time deliveries. Due to its real-world importance, several truck-scheduling works and procedures have been introduced during recent years, treating specific cross-dock settings. For that reason, we highlight in Table 1, some outstanding works dealing with truck scheduling problems in a CDC. As proposed by [15], a classification scheme for deterministic truck scheduling problems is presented for each work, as well as, their contribution.

Table 1: Previous specific research works for the truck scheduling problems. Based on [16].

Publication	Notation	Complexity	Contribution
Miao et al. [41]	$[M limit, t_{io} \star]$	NP-hard	MM, HM, P
Chen and Lee [20]	$[E2 t_j = 0 C_{max}]$	NP-hard	B, ES, P
Chen and Lee [20]	$[E2 t_j = 0 C_{max}]$	NP-hard	P
Chen and Song [21]	$[E t_j = 0 C_{max}]$	NP-hard	MM, HS, B
Boysen [15]	$[E p_j = p, no - wait, t_j = 0 \sum T_o]$	Open	MM, HM, ES
Boysen and Flidner [16]	$[E t_{io}, f_{ix} \sum W_s U_s]$	NP-hard	MM, P
Boysen and Flidner [16]	$[E t_i = 0, f_{ix} \sum W_s U_s]$	NP-hard	P
Boysen et al. [17]	$[E2 p_j = p, change C_{max}]$	NP-hard	MM, HS, HI, ES, P
McWilliams [40]	$[E no - wait \star]$	Open	HM
Vahdani and Zandieh [55]	$[E2 change C_{max}]$	NP-hard	MM, HM
Melo and Araujo [4]	$[E t_j = 0 C_{max}]$	NP-hard	HM
Arabani et al. [3]	$[E2 change C_{max}]$	Open	HM
Larbi et al. [33]	$[E2 pmtn \star]$	NP-hard	MM, ES
Alpan et al. [2]	$[E pmtn \star]$	Open	MM, ES
Lira [39]	$[E t_j = 0 C_{max}]$	NP-hard	HM
Lima [38]	$[E2 t_j = 0 \sum C_j^2]$	NP-hard	MM, HS, HM
Fonseca [26]	$[E2 t_j = 0 C_{max}]$	NP-hard	MM, HI, B
Cota et al. [25]	$[E t_j = 0 C_{max}]$	NP-hard	MM, HS

The notation used in column “Contribution” is stated as: MM (mathematical model), HI (heuristic improvement procedure), HM (meta-heuristic), B (bound computation), HS (start heuristic for initial solution), ES (exact solution procedure), P (properties [e.g., complexity] of problem). In the Boysen’s [15] tuple notation, column “Notation”, the fields are door environment, operational characteristics and the objective, respectively.

Chen and Lee [20] study a two-machine cross-docking flow shop scheduling problem, in which a job at the second machine can be processed only after finalizing some jobs at the first one, with the objective of minimizing the makespan. The authors show the problem is strongly NP-hard. They develop a polynomial approximation algorithm with an error-bound analysis and a branch-and-bound algorithm. Computational results show that the branch-and-bound algorithm can optimally solve problems with up to 60 jobs in a reasonable time.

Chen and Song [21] extended the Chen and Lee [20] problem to a hybrid flow shop by considering multiple parallel processors (multiple docks) per stage

(inbound and outbound), allowing simultaneous loading and unloading operations. They propose a mixed integer programming model, and four constructive heuristics based on Johnson’s rule.

For the sake of comparison, our article deals with Chen and Lee’s [20] two-machine cross-docking flow shop scheduling problem.

The hybrid metaheuristics arise as an option to achieve solutions within a reasonable computational time in very difficult combinatorial optimization problems. The hybrid method consists of a cooperative combination of methods, exact and/or approximated, and aims to absorb to the limit the potentialities of all the approaches (see surveys Alba [1], Puchinger and Raidl [48] and Blum et al. [13]).

Paula et al. [45] propose a variant of the Lagrangean Relaxation to obtain good bounds to the problem of parallel machines scheduling with sequence-dependent setup times. In this effort, the Lagrangean Relaxation is improved by the use of a metaheuristic as an internal procedure of the Lagrangean Relaxation. The heuristic is used to generate feasible solutions during the execution of the Non-delayed relax-and-cut algorithm.

Pirkwieser et al. [47] present a similar work, using the Lagrangean Relaxation applied to the Knapsack Constrained Maximum Spanning Tree problem, in which the Lagrangean Multipliers influence the procedures executed by the Genetic Algorithm.

Boschetti and Maniezzo [14] propose a Lagrangean Metaheuristic procedure. Their approach consists in using metaheuristic techniques to obtain feasible solutions using the information of Lagrangean Multipliers. The solutions obtained tend to be good bounds. We remark that, in these efforts, the potential of the information of the Lagrangean Multipliers is still not completely used. Part of our goals is to improve the use of this information. Readers are referred to [13] for a survey on hybrid Metaheuristics.

As discussed here, a significant number of articles on CDC are studying strategical and tactical aspects. However, not always aligned with industry practice. Ladier and Alpan [32] discuss the relationship between the industry

needs and academic research topics, pointing out the gap between both worlds.

2.1. Contributions of this article

In this manuscript, we propose a Hybrid Lagrangean Metaheuristic Framework for the two-machine cross-docking flow shop scheduling problem [20]. The work has a similar line of thought than Boschetti and Maniezzo [14], however, it emphasizes the use of the information of the Lagrangean Multipliers as discussed in Pirkwieser et al. [47], and Paula et al. [45]. Although, the subproblem of the Lagrangean relaxation is polynomial, through a series of cuts on the makespan value, we improve the Linear relaxation lower bound limits. The proposed algorithm uses the information obtained from the Lagrangean multipliers to construct feasible solutions through the heuristic NEH ([42]), and performs search procedure through a Local Search framework. Furthermore, the proposed hybrid algorithm proves optimality in several instances.

3. Mathematical Model

In this section, we present a mathematical formulation based on scheduling problem initially proposed by Chen and Lee [20]. The integer programming model considered in this work adopts a time-indexed formulation proposed by Lima [38] and deals with the objective function as proposed by Chen and Lee [20].

Considering a cross-docking center where n loaded trucks arrive with products demanded by one or more customers. Each truck must unload its cargo and load it into m output trucks, responsible for delivery to specific destinations in the supply chain. Each output truck may leave the center only after the truck is fully loaded, and it can begin the loading process only after all needed products were unloaded in the center. Here we consider the existence of two docks in the center, one dock dedicated to unload the trucks and the other one for loading purposes, machine 1 (M1) and machine 2 (M2), respectively. The problem is to define the sequence to process inbound and outbound trucks minimizing the

completion time of the last job processed by machine M2, commonly known as makespan.

The following sets, parameters and variables are considered:

Sets

The set of periods is defined as $T = \{0, \dots, T_f\}$. To represent arrivals and truck departures in cross-docking center, two sets of jobs are created:

- $T = \{0, \dots, T_f\}$, set of discrete periods considered.
- $J^1 = \{j_1^1, j_2^1, \dots, j_n^1\}$, set of inbound trucks, which must be processed on M1.
- $J^2 = \{j_1^2, j_2^2, \dots, j_m^2\}$, set of outbound trucks, which must be processed on M2.
- S_j , set of precedent jobs. For each job $j_j^2 \in J^2$, there is a corresponding subset of J^1 that must be completed before beginning its process. It is considered that each element job $j \in J^2$ has at least one job S_j as precedent.

Input Parameters

- n : number of jobs to be processed on M1.
- m : number of jobs to be processed on M2.
- p_{ij} : processing time of job j on machine i .
- T_f : size of time horizon. A first estimate for the time horizon is the sum of the processing times of all jobs.
- $T_0 \geq 0$: beginning of time horizon for the jobs $j \in J^2$, i.e., minimum starting date for processing of the output trucks, initially set as $T_0 = \min_{j \in J^2} \{\sum_{i \in J^1, i \in S_j} p_{1i}\}$, then $T = \{T_0, \dots, T_f\} \forall j \in J^2$.

Decision variables

- x_{jt} ($\forall j \in J^1, \forall t \in T$), binary variable, x_{jt} is equal to 1 if job j starts its process at time t and equal to 0 otherwise.
- y_{jt} ($\forall j \in J^2, \forall t \in T$), binary variable, y_{jt} is equal to 1 if job j starts its process at time t and equal to 0 otherwise.

The complete mathematical model (F) is presented as follows:

$$(F) \quad \text{Minimize } C_{max} \quad (1)$$

subject to

$$\sum_{t=0}^{T_f-p_{1j}} x_{jt} = 1, \quad \forall j \in J^1, \quad (2)$$

$$\sum_{t=T_0}^{T_f-p_{2j}} y_{jt} = 1, \quad \forall j \in J^2, \quad (3)$$

$$\sum_{j \in J^1} \sum_{s=\max(0; t-p_{1j}+1)}^t x_{js} \leq 1, \quad \forall t \in T, \quad (4)$$

$$\sum_{j \in J^2} \sum_{s=\max(T_0; t-p_{2j}+1)}^t y_{js} \leq 1, \quad \forall t \in T, \quad (5)$$

$$\sum_{t=T_0}^{T_f-p_{2j}} t y_{jt} - \sum_{t=0}^{T_f-p_{1k}} (t + p_{1k}) x_{kt} \geq 0, \quad \forall j \in J^2, \forall k \in S_j, \quad (6)$$

$$C_{max} \geq p_{2j} + \sum_{t=T_0}^{T_f-p_{2j}} t y_{jt}, \quad \forall j \in J^2, \quad (7)$$

$$x_{jt} \in \{0, 1\}, \quad \forall j \in J^1, \forall t \in T, \quad (8)$$

$$y_{jt} \in \{0, 1\}, \quad \forall j \in J^2, \forall t \in T, \quad (9)$$

$$C_{max} \geq 0. \quad (10)$$

The objective function (3) minimizes the makespan. The set of constraints (2) ensures that each job $j_j^1 \in J^1$ should start its processing in one and only one period within the time horizon. The set (3) works with the same reasoning

on M2. Constraints (4) ensure that a job $j_j^1 \in J^1$ does not start its processing while another job is being processed in the same machine M1. The constraints set (5) works, in the same way, but applied to jobs $j_j^2 \in J^2$. The set (6) controls the precedence relationships. For every existing precedence relation, the start date of the job $j_j^2 \in J^2$ should be greater than or equal to the completion time of its precedent $j_k^1 \in S_j$. The set of constraints (7) indicates that the variable C_{max} should be the maximum completion time of jobs in J^2 . Finally sets (8) to (10) define the variable's domain.

3.1. Lagrangean Relaxation

Let $L(\lambda)$ be the relaxation of formulation F, when constraints 6 are dualized and its violation penalized in the objective function. For each constraint is associated one Lagrangean multiplier λ representing the weight given to the violation. The set of Lagrangean multipliers will be represented by λ_{jk} , with $j \in J^2$ and $k \in S_j$. Notice that, these constraints are the cross-docking precedence relations, coupling decisions in M1 with decisions in M2. Thus the subproblem $L(\lambda)$ is defined as:

$$L(\lambda) = \text{Minimize } C_{max} + \sum_{j \in J^2} \sum_{k \in S_j} \lambda_{jk} \left(\sum_{t=0}^{T_f - p_{1k}} (t + p_{1k}) x_{kt} - \sum_{t=T_0}^{T_f - p_{2j}} t y_{jt} \right) \quad (11)$$

$$\text{s.t. } (2) - (5), (7) - (10), \lambda \geq 0.$$

Now we are able to uncouple the problem into two new scheduling problems, one for each machine. Subproblem X considers M1 while subproblem Y considers M2. The value of the lower bound is obtained by adding the subproblems objective functions $L(\lambda)_x$ and $L(\lambda)_y$.

Subproblem X

Isolating the terms of the objective function that contain variables x , we reach to :

$$L(\lambda)_x = \text{Minimize } \sum_{j \in J^2} \sum_{k \in S_j} \lambda_{jk} \sum_{t=0}^{T_f - p_{1k}} (t + p_{1k}) x_{kt} \quad (12)$$

$$\text{s.t. } (2), (4) \text{ and } (8).$$

Rewriting the above objective function from the perspective of jobs in J^1 , we have:

$$L(\lambda)_x = \text{Minimize} \sum_{j \in J^1} \sum_{t=0}^{T_f - p_{1j}} (t + p_{1j}) x_{jt} w_j^1 \quad (13)$$

where $w_j^1 = \sum_{i \in J^2} \lambda_{ij}$, where $\lambda_{ij} = 0$ if $j \notin S_i$.

The problem mentioned above is known as The Total Weighted Completion Time ([46]), denoted by $1||\sum C_j W_j$. This problem can be solved by the WSPT rule (Weighted Shortest Processing Time First) proposed by Smith in 1956 [50]. According to this rule, the optimal solution is obtained by ordering the jobs in descending order of w_j^1 / p_{1j} , and can be obtained in $O(n \log(n))$.

Subproblem Y

With the terms associated to variables y , subproblem Y can be defined as:

$$L(\lambda)_y = \text{Minimize} \quad C_{max} - \sum_{j \in J^2} \sum_{k \in S_j} \lambda_{jk} \sum_{t=T_0}^{T_f - p_{2j}} t y_{jt} \quad (14)$$

subject to

$$\sum_{t=T_0}^{T_f - p_{2j}} y_{jt} = 1, \quad \forall j \in J^2, \quad (15)$$

$$\sum_{j \in J^2} \sum_{s=\max(T_0; t-p_{2j}+1)}^t y_{js} \leq 1, \quad \forall t \in T, \quad (16)$$

$$C_{max} \geq p_{2j} + \sum_{t=T_0}^{T_f - p_{2j}} t y_{jt}, \quad \forall j \in J^2, \quad (17)$$

$$y_{jt} \in 0, 1, \quad \forall j \in J^2, \forall t \in T, \quad (18)$$

$$\lambda_{jk} \geq 0, \quad \forall j \in J^2, k \in S_j, \quad (19)$$

$$C_{max} \geq 0. \quad (20)$$

Defining $w_j^2 = -\sum_{k \in S_j} \lambda_{jk}$ $\forall j \in J^2$, we can rewrite the problem as follows:

$$L(\lambda)_y = \text{Minimize } C_{max} + \sum_{j \in J^2} \sum_{t=T_0}^{T_f - p_{2j}} t \cdot y_{jt} w_j^2 \quad (21)$$

s.t. (15), (16), (17), (18) e (20).

The first term of the objective function (21) is the makespan. The second term represents the sum of the weighted starting times of the jobs on M2, also known as The Total Weighted Starting Time, denoted $\sum I_j W_j$. Thus, the Y subproblem can be defined as $1||C_{max} + \sum I_j W_j$. To solve this subproblem, we proposed a rule named WSPT-TRD.

The WSPT-TRD rule works as follows. First, the WSPT rule is used to generate a sequence of jobs on M2. The second step is to define the correct allocation of jobs in the time horizon. For that, we evaluate the increase of the makespan when the weights associated with the jobs are negative. When the weights are negative, it creates a trade-off situation, as explained next.

As previously mentioned, weights w_j^2 can have positive or negative values. Let us first consider schedule decisions for jobs with $w_j^2 \geq 0$ (in our case $w_j^2 = 0$, given that $\lambda_{jk} \geq 0$), both C_{max} and $\sum I_j W_j$ have similar behavior, since the objective function is minimized by allocating the jobs at the beginning of the time horizon. In our particular case, as $\lambda_{jk} \geq 0$, we do not have the case of $w_j^2 > 0$, however our algorithm considers this case.

Considering now jobs with negative weights ($w_j^2 < 0$), the objective function terms lead to different solutions. On the one hand, by considering the makespan C_{max} one tends to allocate the jobs at the beginning of time horizon, as the weights do not influence its final value. On the other hand, when analyzing the $\sum I_j W_j$ one tends to allocate the jobs as late as possible, taking advantage of the negative weights.

We notice that if we delay all jobs with negative weights one-time unit, the C_{max} increases one unit. Thus, our algorithm computes the sum of the negative weights, if the sum of the negative weights is less than -1 , the displacement of the jobs towards the end of the time horizon compensates the increase of C_{max} , defining in this way the best allocation of jobs.

Pseudocode of algorithm WSPT-TRD:

Step 1: Compute the weights w_j^2 on the machine 2 ($w_j^2 = -\sum_{k \in S_j} \lambda_{jk} \forall j \in J^2$).

Step 2: Sort the jobs in J^2 in decreasing order of w_j^2 / p_{2j} .

Step 3: Compute the sum of the negative weights as $\delta = \sum_{j \in J^2} w_j^2 \forall j \in J^2 \wedge w_j^2 < 0$.

Step 4: If $w_j^2 = 0$ or $\delta \geq -1$, allocate the jobs from $t = T_0$ by sequence generated by WSPT rule.

Step 5: If $w_j^2 < 0$ and $\delta \leq -1$, allocate the jobs from $t = T_f$ by the reverse sequence generated by WSPT rule.

Theorem 1. *Considering the problem 1|| $C_{max} + \sum I_j W_j$, the algorithm WSPT-TRD obtains an optimal solution.*

Proof 1. *The proof is provided in Appendix A.*

3.2. Lower Bounds

Besides the lower bound obtained from the Lagrangean relaxation, we consider here two other lower bounds. The first one, called LB_1 is defined as proposed by Chen and Lee [20], as follows:

$$LB_1 = \sum_{i \in J^2} p_{2i} + \min_{j \in J^2} \left\{ \sum_{i \in J^1, i \in S_j} p_{1i} \right\}.$$

LB_1 basically computes the sum of processing time on $M2$, plus the minimum amount of time necessary to begin the process of the first job in $M2$, that is, the minimum set of precedence jobs in $M1$.

The second lower bound, LB_2 , proposed in this article, considers a complementary condition:

$$LB_2 = \sum_{i \in J^1} p_{1i} + \min_{j \in J^1} \left\{ \sum_{i \in J^2, i \in P_j} p_{2i} \right\}, \text{ with } P_j \text{ the successors of } j \in J^1.$$

In this case, LB_2 computes the sum of processing times in $M1$ plus the minimum set of common successors of $j \in J^1$. Finally, the Lower Bound is defined as the maximum lower bound, $LB = \max\{LB_1, LB_2\}$.

It is worth noticing that with a valid LB we can redefine a new date for the beginning of the processing time of jobs in the $WSPT - TRD$ algorithm. Particularly, in *Step 4*, $T_0 = LB - \sum_{i \in J^2} p_{2i}$, this is valid for the jobs with $w_j^2 = 0$ or for all jobs when $\delta \geq -1$.

4. Hybrid Lagrangean Metaheuristic

In the Hybrid Lagrangean Metaheuristic Framework (HgR) proposed in this work, the Lagrangean Dual is solved by the Volume algorithm, as proposed in [7] and [43]. Readers are referred to [8] and [28] for a discussion on the Volume algorithm and its performance. In our case, the Lagrangean Relaxation incorporates heuristics as internal procedures with focus in obtaining feasible solutions. These heuristics are based on a constructive method and Local Search. Both procedures are executed sequentially under the Lagrangean Relaxation.

The Volume algorithm is an extension of the subgradient algorithm, which will produce a sequence of primal and dual solutions, thus being able to prove optimality. This algorithm has similar computational effort than the subgradient algorithm and it presents similarities with the Conjugate Subgradient method [35], [57] and the Bundle method [36, 37]. A discussion of its main features and a global convergence analysis can be found in [6].

In the proposed algorithm the precedence constraints are dualized. The Lagrangean multipliers define a penalty to a given job allocated at a given position, if the job has a large associated value, it means that it has greater impact on the objective function, i.e., it is allocated in a disadvantageous position. This information can be used to decide when to schedule the jobs.

Let x and y be solutions of subproblems X and Y , respectively, with objective function value z . Then λ are the Lagrangean multipliers obtained by the Volume Algorithm and $\nu(\lambda, x, y)$ is the subgradient. Let UB be the upper

bound or feasible solution. The initial UB is generated by a Lagrangean meta-heuristic. The flow diagram of the proposed algorithm is depicted in Figure 2 and its steps can be summarized as:

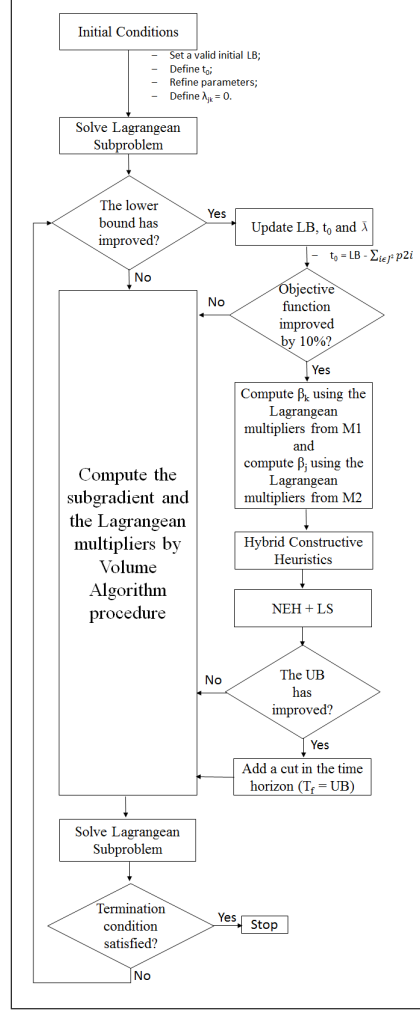


Figure 2: Data flow chart of the proposed methodology.

Step 0: We start with a null vector λ_{jk} ($\lambda_{jk}=0$). Solve the Lagrangean subproblem to obtain the initial solution of X subproblem (x^0), Y subproblem (y^0) and objective function value (z^0). Let UB be the feasible solution ob-

tained by Lagrangean metaheuristic. Set $\bar{x} = x^0$, $\bar{y} = y^0$, $\bar{z} = z^0$ and $k=1$.

Step 1: Compute the subgradient $\nu(\lambda_{jk-1}, \bar{x}, \bar{y})$ and $\lambda_{jk} = \bar{\lambda}_{jk} + s\nu(\lambda_{jk-1})$, the calculation of the step size s is given by the equation (25). Solve the Lagrangean subproblem with the new λ_{jk} and let x^k , y^k and z^k be the solutions obtained. Then $\bar{x} = \alpha x^k + (1-\alpha)\bar{x}$, $\bar{y} = \alpha y^k + (1-\alpha)\bar{y}$ and $\bar{\lambda} = \alpha \lambda^k + (1-\alpha)\bar{\lambda}$, where α is a number between 0 and 1, defined by convex combination such as defined in (22).

Step 2: If $z^k > \bar{z}$ update $\bar{\lambda} = \lambda_{jk}$ and $\bar{z} = z^k$. If \bar{z} improves by 10% since the last run of the Lagrangean metaheuristic then go to Step 3. Else go to Step 4.

Step 3: Lagrangean metaheuristic:

1. list \leftarrow Hybrid Constructive Heuristics (H1 and H2);
2. $UB^k \leftarrow$ NEH (list);
3. $UB^k \leftarrow$ Local Search (list);

If $UB^k < UB$ update $UB = UB^k$ and add a cut in order to reduce the time horizon, improving the limits found (update $T_f = UB$).

Step 4: Stop criteria: If satisfied stop. Else let $k = k + 1$ and go to Step 1.

The step size s and the parameter α used to define \bar{x} and \bar{y} , are computed as proposed in [7] and [28]. First, we define α_{opt} as:

$$\alpha_{opt} = \underset{\alpha}{\operatorname{argmin}} \left\| \alpha \nu'_{(\lambda_{jk-1}, x^k, y^k)} + (1-\alpha) \nu_{(\lambda_{jk-1}, \bar{x}, \bar{y})}^k \right\|^2 \quad (22)$$

The parameter values π , $MaxWaste$, $factor$, α_{max} , st , α , *yellow* and *green*, some of them yet to be introduced, were defined using the SPOT method, as explained in Appendix B. The parameters α_{max} and α are initially defined as 0.3034 and 0.0830 respectively, based on computational tests performed by SPOT. And α is computed as:

$$\alpha = \alpha_{max} * \alpha \quad \text{if} \quad \alpha_{opt} < 0 \quad (23)$$

$$\alpha = \min\{\alpha_{opt}, \alpha_{max}\} \quad \text{if} \quad \alpha_{opt} \geq 0 \quad (24)$$

Before setting the value of the step s , we need to define the parameter π . Therefore, to set the value of π we define three types of algorithms iterations as defined in [7] and [8].

Each iteration with no improvement is named *red*. Then, the parameter *MaxWaste* represents the maximum number of iterations without a lower bound improvement. If $z^k > \bar{z}$ and $\nu'_{(\lambda_{jk-1}, x^k, y^k)} \nu_{(\lambda_{jk-1}, \bar{x}, \bar{y})}^k < 0$, it means that a longer step in the direction to ν^k would have given a smaller value for z^k . Those iterations are denominated *yellow*, otherwise, the iteration is denominated *green*. At each *yellow* iteration we would multiply π by the *yellow* parameter. At each *green* iteration we would multiply π by the *green* parameter. After a sequence of 24 consecutive *red* iterations, we would multiply π by *factor* parameter. Thus, the step size s at iteration k is defined as:

$$s = \frac{\pi * (st * UB - \bar{z})}{\|\nu_{(\lambda_{jk-1}, \bar{x}, \bar{y})}^k\|^2} \quad (25)$$

In the “Lagrangean Methaheuristic step” two hybrid constructive heuristics, the heuristic NEH, [42] and Local Search are executed sequentially. The heuristics are described in the next subsection. In the proposed implementation we use the information obtained from the Lagrangean multipliers to construct feasible solutions through the heuristic NEH and then, we perform an improvement step through a Local Search procedure.

The proposed NEH heuristic uses a “list” L with m jobs generated by the hybrid constructive heuristics (H1 and H2). This “list” L is a sequence of jobs on machine 2. Based on Property 1 in [20] we can generate an optimal sequence on machine 1. At each step, the job at the beginning of the list is inserted in the best position, according to the objective function. The NEH heuristic is applied to the whole list generating a feasible solution.

The Local Search is implemented based on the proposals of [5] and [52]. The procedure adopted is composed by “swap” and “insertion” moves in the

sequence on machine 2 generated by NEH. The former consists of interchanging all pairs of jobs. The latter consists of removing a job from its original position and inserting it on one of the $n - 1$ remaining positions. The local search procedure stops when it is unable to improve the solution further.

Finally, the stop criterion is determined if one of the following criteria is satisfied:

1. Maximum number of iterations, in this case 1000 iterations, or;
2. Relative tolerance GAP defined as: $\frac{z^k - \bar{z}}{\bar{z}} < 0.1$, or;
3. Null Subgradient module or negligible: $\| \nu_{(\lambda_{jk-1}, \bar{x}, \bar{y})} \|^2 \leq 0.000001$.

4.1. Hybrid Constructive Heuristics

In this section, we describe two heuristics used to obtain feasible solutions. The heuristic H1 uses information from Lagrangean multipliers to sort jobs on M1, while H2 uses information from Lagrangean multipliers to sort jobs on M2. The two heuristics are described below:

H1

Step 1: For each job on the machine 1 ($k \in J^1$), compute $\beta_k = \frac{\sum_{j \in J^2} \sum_{k \in J^1} \lambda_{jk}}{Nprec_k}$.

Where $Nprec_k$ corresponds to the number of jobs in the second stage that are waiting for the completion of job k on M1.

Step 2: Get the sequence on M1 by ordering jobs in decreasing order of β_k .

Step 3: Calculate the new release dates ($r_j, j \in J^2$) of jobs on M2.

Step 4: Get the sequence on M2 by ordering jobs in non-decreasing order of release dates r_j .

H2

Step 1: For each job on M2 ($j \in J^2$), compute $\beta_j = \frac{\sum_{j \in J^2} \sum_{k \in S_j} \lambda_{jk}}{Nprec_j}$.

Where $Nprec_j$ corresponds to the number of precedents that job j has.

Step 2: Get the sequence on M2 by ordering jobs in increasing order of β_j .

Step 3: Sequence the jobs on the M1 according the sequence on M2, respecting the precedence relations of cross-docking.

If there is a job on M1 without precedence relationship in the machine 2, schedule this job last.

Step 4: Calculate the new release dates ($r_j, j \in J^2$) of jobs on M2, from the sequence on M1.

Step 5: Get the sequence on M2 by ordering jobs in non-decreasing order of release dates r_j .

5. Computational Experiments

To investigate the performance of the Complete Model, Linear Relaxation and Hybrid Lagrangean Metaheuristic Framework, artificial instances are generated varying the processing time of jobs and the number of jobs on machines 1 and 2, according to Chen and Lee [20]. Tests are run on a Intel (R) Xeon (R) CPU X5690 @ 3.47GHz with 24 processors, 132 GB memory and Linux operational system. The programming language used is C++ with the optimization software CPLEX 12.4.

5.1. Instances Generation

The instances used in this work are generated through the software MATLAB, following the description found in [20].

Table 2 presents a summary of the generated instances and its characteristics. All processing times are randomly defined, as well as its predecessors, in accordance with the uniform distribution shown in the table. The number of jobs in each stage (n and m) is defined a priori. For each n , five instances with different m values are considered. And for each pair (n, m) , we generate 10 different problems, therefore the benchmark consists of 500 instances, 50 instances

Table 2: Summary of the artificial benchmark. It is divided into two groups, each row informs the features of a sub-group of instances. n and m indicate the number of jobs in each stage. (NP) the uniform distribution to select the number predecessors of jobs $j \in J^2$ and (TP) the distribution of processing time for all jobs.

Group	Jobs	Jobs	NP	TP
	M1(n)	M2(m)		
1	5	3-4-5-6-7	U(1,4)	U(1,10)
	10	6-8-10-12-14	U(1,9)	U(1,10)
	20	12-16-20-24-28	U(1,19)	U(1,10)
	40	24-32-40-48-56	U(1,39)	U(1,10)
	60	36-48-60-72-84	U(1,59)	U(1,10)
2	5	3-4-5-6-7	U(1,4)	U(10,100)
	10	6-8-10-12-14	U(1,9)	U(10,100)
	20	12-16-20-24-28	U(1,19)	U(10,100)
	40	24-32-40-48-56	U(1,39)	U(10,100)
	60	36-48-60-72-84	U(1,59)	U(10,100)

per row of Table 2 ¹.

5.2. Computational Results

We compare the performance of the Complete Model (MIP), the Linear Relaxation (LR) and Hybrid Lagrangean Metaheuristic Framework (HgR). Results presented in Table 3 depict the average of 10 cases for each instance size (n , m).

Columns n and m show respectively the number of jobs in the first and second machine. Column LB refers to the lower bound, column UB refers to the upper bound, column $GAP(\%)$ is characterized as $\frac{(Upper\ Bound - Lower\ Bound)}{Upper\ Bound}$, for instances where no feasible solution are found, it is considered $GAP=100\%$. The column $T(s)$ refers to CPU time expended to solve the problem in seconds, the column $\% Sol.$ represents the percentage of problems with at least one fea-

¹all instances are available at <https://sites.google.com/site/martingravetti/>

sible solution found, the column *% Solv. at Opt.* represents the percentage of problems to solve at optimality, the column *% Solved* represents the percentage of problems solved by the linear relaxation, column UB_{H1} refers to the best upper bound found by H1, in the same way, column UB_{H2} refers to the best upper bound found by H2.

The run time is limited to one hour of CPU time (3.600 seconds), and results are depicted in Table 3 are registered. The dash (-) means that the corresponding value is not found. The linear programming relaxation *GAP* is defined as the relative difference between the upper bound found by MIP and the lower bound found by the LR. In addition, the *GAP* of the HgR is calculated considering the best upper bound between UB_{H1} and UB_{H2} , and its lower bound. Finally, after comparing and analyzing the MIP, LR and HgR the best lower and upper bounds obtained are highlighted in the table.

Table 3: Average computational results for complete model, linear relaxation and hybrid lagrangean metaheuristic framework.

Group 1 - Processing time [1, 10]																
n	m	MIP						LR				HgR				
		LB	UB	GAP	T(s)	% Sol.	% Solv. at Opt.	LB	GAP	T(s)	% Solved	LB	UB _{H1}	UB _{H2}	GAP	T(s)
5	3	33.0	<u>33.0</u>	0.0%	0.2	100.0%	100.0%	23.1	30.0%	0.0	100.0%	31.5	<u>33.0</u>	<u>33.0</u>	4.8%	0.0
	4	33.0	<u>33.0</u>	0.0%	0.3	100.0%	100.0%	23.1	29.5%	0.0	100.0%	30.7	<u>33.0</u>	<u>33.0</u>	5.9%	0.0
	5	34.3	<u>34.3</u>	0.0%	0.5	100.0%	100.0%	23.0	32.9%	0.0	100.0%	33.4	<u>34.3</u>	34.6	2.5%	0.0
	6	39.1	<u>39.1</u>	0.0%	1.3	100.0%	100.0%	24.1	37.8%	0.0	100.0%	37.0	39.2	39.3	5.2%	0.0
	7	43.5	<u>43.5</u>	0.0%	3.7	100.0%	100.0%	25.0	42.2%	0.0	100.0%	40.9	43.9	<u>43.5</u>	5.4%	0.0
	Subgroup Average	36.6	<u>36.6</u>	0.0%	1.2	100.0%	100.0%	23.7	34.5%	0.0	100.0%	34.7	36.7	36.7	4.8%	0.0
10	6	58.7	<u>58.7</u>	0.0%	20.2	100.0%	100.0%	35.6	39.1%	0.0	100.0%	57.7	58.8	<u>58.7</u>	1.7%	0.0
	8	65.7	<u>65.7</u>	0.0%	167.5	100.0%	100.0%	38.5	41.4%	0.1	100.0%	62.7	66.6	66.2	5.4%	0.0
	10	70.9	<u>70.9</u>	0.0%	369.6	100.0%	100.0%	39.2	44.7%	0.1	100.0%	66.8	71.4	71.3	6.4%	0.0
	12	77.3	<u>81.6</u>	4.9%	2562.5	100.0%	30.0%	40.2	50.4%	0.1	100.0%	77.1	81.7	<u>81.6</u>	5.3%	0.0
	14	86.0	<u>90.1</u>	4.5%	3242.3	100.0%	10.0%	43.8	51.4%	0.1	100.0%	85.5	92.0	92.9	7.0%	0.0
	Subgroup Average	71.7	<u>73.4</u>	1.9%	1227.9	100.0%	68.0%	39.5	45.4%	0.1	100.0%	70.0	74.1	74.1	5.2%	0.0
20	12	114.1	134.8	14.8%	3600.0	100.0%	0.0%	65.3	51.4%	0.3	100.0%	123.6	<u>132.3</u>	133.0	6.6%	0.0
	16	103.5	152.6	31.3%	3600.0	100.0%	0.0%	65.7	56.6%	0.7	100.0%	132.4	<u>146.9</u>	147.9	9.8%	0.0
	20	106.6	167.3	35.8%	3600.0	100.0%	0.0%	68.0	59.2%	0.6	100.0%	141.3	158.8	<u>158.5</u>	10.9%	0.0
	24	115.6	189.4	38.2%	3600.0	100.0%	0.0%	72.5	61.4%	0.9	100.0%	154.9	<u>178.7</u>	180.9	12.8%	0.0
	28	129.6	205.9	36.5%	3600.0	100.0%	0.0%	81.9	59.9%	1.0	100.0%	164.8	194.9	<u>193.9</u>	15.0%	0.0
	Subgroup Average	113.9	167.1	31.3%	3600.0	100.0%	0.0%	70.7	57.7%	0.7	100.0%	143.4	<u>162.3</u>	162.8	11.0%	0.0
40	24	164.1	306.1	46.3%	3600.0	100.0%	0.0%	122.3	60.0%	2.0	100.0%	256.1	283.7	<u>281.4</u>	9.0%	0.0
	32	162.4	376.0	56.7%	3600.0	100.0%	0.0%	125.5	66.6%	3.5	100.0%	276.7	317.0	<u>315.9</u>	12.5%	0.0
	40	161.8	421.3	69.2%	3600.0	80.0%	0.0%	125.9	70.0%	7.5	100.0%	297.0	339.6	<u>338.9</u>	12.3%	0.0
	48	-	-	-	3600.0	0.0%	0.0%	134.5	-	12.7	100.0%	315.3	<u>378.5</u>	379.2	16.6%	0.0
	56	-	-	-	3600.0	0.0%	0.0%	156.1	-	26.2	100.0%	329.3	410.4	<u>408.5</u>	19.3%	0.0
	Subgroup Average	*	*	*	3600.0	56.0%	0.0%	132.9	*	10.4	100.0%	294.9	345.8	<u>344.8</u>	13.9%	0.0
60	36	-	-	-	3600.0	0.0%	0.0%	176.2	-	34.5	100.0%	395.3	432.9	<u>431.9</u>	8.4%	0.0
	48	-	-	-	3600.0	0.0%	0.0%	176.3	-	43.0	100.0%	415.5	<u>472.3</u>	476.5	12.0%	0.0
	60	-	-	-	3600.0	0.0%	0.0%	181.1	-	133.1	100.0%	438.1	<u>535.5</u>	535.9	18.2%	0.0
	72	-	-	-	3600.0	0.0%	0.0%	202.8	-	112.7	100.0%	473.5	585.5	<u>584.5</u>	18.9%	0.0
	84	-	-	-	3600.0	0.0%	0.0%	240.5	-	122.8	100.0%	499.0	639.1	<u>635.5</u>	21.4%	0.0
	Subgroup Average	*	*	*	3600.0	0.0%	0.0%	195.4	*	89.2	100.0%	444.3	533.1	<u>532.9</u>	16.0%	0.0
Group 2 - Processing time [10, 100]																
5	3	317.3	<u>317.3</u>	0.0%	21.0	100.0%	100.0%	225.5	29.0%	0.2	100.0%	310.5	<u>317.3</u>	<u>317.3</u>	2.3%	0.0
	4	329.8	<u>329.8</u>	0.0%	354.9	100.0%	100.0%	228.5	30.4%	0.3	100.0%	306.6	<u>329.8</u>	<u>329.8</u>	6.2%	0.0
	5	339.3	<u>342.7</u>	0.8%	1570.1	100.0%	70.0%	226.7	33.7%	0.4	100.0%	333.9	<u>342.7</u>	<u>342.7</u>	2.6%	0.0
	6	339.9	393.3	11.8%	2520.0	100.0%	40.0%	237.0	39.3%	0.6	100.0%	372.0	<u>392.2</u>	398.5	4.8%	0.0
	7	379.3	441.2	12.2%	2965.3	100.0%	20.0%	246.5	43.9%	1.7	100.0%	412.2	<u>436.7</u>	445.0	5.1%	0.0
	Subgroup Average	341.1	364.9	5.0%	1486.3	100.0%	66.0%	232.8	35.3%	0.7	100.0%	347.0	<u>363.7</u>	366.7	4.2%	0.0
10	6	437.7	622.2	28.8%	3590.1	100.0%	10.0%	355.2	42.7%	9.8	100.0%	580.4	<u>589.1</u>	<u>589.1</u>	1.4%	0.0
	8	455.7	728.8	36.1%	3600.0	100.0%	0.0%	362.8	49.5%	10.9	100.0%	641.1	<u>667.5</u>	670.8	3.7%	0.0
	10	443.0	819.5	45.2%	3600.0	100.0%	0.0%	375.3	53.8%	12.5	100.0%	689.5	<u>747.2</u>	750.6	7.6%	0.0
	12	445.9	997.9	63.3%	3600.0	80.0%	0.0%	393.2	60.7%	29.7	80.0%	771.1	<u>817.0</u>	817.2	5.4%	0.0
	14	414.8	1096.9	72.8%	3600.0	70.0%	0.0%	409.1	61.7%	65.9	70.0%	852.2	912.8	<u>905.0</u>	5.8%	0.0
	Subgroup Average	439.4	853.1	49.2%	3598.0	90.0%	2.0%	379.1	54.0%	27.1	90.0%	706.9	746.7	<u>746.5</u>	4.8%	0.0
20	12	-	-	-	3600.0	0.0%	0.0%	651.4	-	195.4	100.0%	1241.5	<u>1326.9</u>	1327.5	6.4%	0.0
	16	-	-	-	3600.0	0.0%	0.0%	654.7	-	237.7	100.0%	1327.8	<u>1445.7</u>	1482.4	8.2%	0.0
	20	-	-	-	3600.0	0.0%	0.0%	674.2	-	322.6	100.0%	1410.4	1590.8	<u>1574.0</u>	10.3%	0.0
	24	-	-	-	3600.0	0.0%	0.0%	720.2	-	453.4	100.0%	1547.2	<u>1770.2</u>	1784.4	12.2%	0.0
	28	-	-	-	3600.0	0.0%	0.0%	828.0	-	1697.9	100.0%	1685.3	1974.1	<u>1955.4</u>	13.5%	0.0
	Subgroup Average	*	*	*	3600.0	0.0%	0.0%	705.7	*	581.4	100.0%	1442.4	<u>1621.5</u>	1624.7	10.1%	0.0
40	24	-	-	-	3600.0	0.0%	0.0%	1183.5	-	3600.0	90.0%	2273.7	2426.0	<u>2409.3</u>	5.6%	0.0
	32	-	-	-	3600.0	0.0%	0.0%	1176.3	-	3600.0	90.0%	2741.6	3114.8	<u>3082.5</u>	11.0%	0.0
	40	-	-	-	3600.0	0.0%	0.0%	1020.5	-	3600.0	70.0%	2954.5	<u>3408.4</u>	3421.8	13.3%	0.0
	48	-	-	-	3600.0	0.0%	0.0%	1279.7	-	3600.0	80.0%	3152.8	3782.0	<u>3759.5</u>	16.1%	0.0
	56	-	-	-	3600.0	0.0%	0.0%	-	-	3600.0	0.0%	3293.4	4053.9	<u>4022.5</u>	18.1%	0.0
	Subgroup Average	*	*	*	3600.0	0.0%	0.0%	*	*	3600.0	66.0%	2883.2	3357.0	<u>3339.1</u>	12.8%	0.0
60	36	-	-	-	3600.0	0.0%	0.0%	-	-	3600.0	0.0%	3960.9	4328.7	<u>4288.7</u>	7.6%	0.0
	48	-	-	-	3600.0	0.0%	0.0%	-	-	3600.0	0.0%	4166.0	<u>4691.7</u>	4736.5	11.2%	0.0
	60	-	-	-	3600.0	0.0%	0.0%	-	-	3600.0	0.0%	4388.6	<u>5279.0</u>	5284.4	16.9%	0.4
	72	-	-	-	3600.0	0.0%	0.0%	-	-	3600.0	0.0%	4745.2	5802.5	<u>5767.5</u>	17.6%	1.4
	84	-	-	-	3600.0	0.0%	0.0%	-	-	3600.0	0.0%	5000.7	6366.1	<u>6291.9</u>	20.5%	2.4
	Subgroup Average	*	*	*	3600.0	0.0%	0.0%	*	*	3600.0	0.0%	4452.3	5293.6	<u>5273.8</u>	14.8%	0.8

The results show that the proposed model is efficient to solve to optimality instances with $n = 5$ and $n = 10$ jobs for group 1 and $n = 5$ jobs for group 2. That is, only 8 of 25 instances of group 1, and only 2 of 25 instances of group 2, which means that the complete model finds the optimal solution in only 20% of the tested instances. Furthermore, for the larger instances of group 1 ($n = 40$ and $n = 60$ jobs) and most of instances of group 2 (from $n=20$ jobs) the model found no solution. These results exposes the difficulty of solving time-indexed models. In the time-indexed problems, the number of variables is proportional to the time horizon, and the higher the number of jobs the greater the horizon of time to sequence them. These factors increase the computational effort to solve these problems.

From the computational results, it is clear that the linear relaxation solves the problems much faster than the MIP but with poor results. The relaxed problem starts with a GAP of approximately 30% for small instances and goes up to 70% for large instances. Thus, it highlights the weakness of the linear relaxation.

When comparing lower bounds, the Hybrid Lagrangean Metaheuristic framework provides better bounds than the linear relaxation. The HgR found better lower bounds in all instances (100% of the instances). Furthermore, the HgR solved all the instances almost instantaneously, while the linear relaxation consumed more time, not being able to find solutions for larger instances of group 2. For the cases in which the complete model showed GAP higher than zero, the lower bound of the Hybrid Lagrangean Metaheuristic Framework it was better for most cases. It is also worth mentioning that in almost all cases in which the complete model does not found the optimal solution the GAP of the HgR was better than the GAP of the complete model (94% of all tested instances).

The increase in the number of jobs results in a significant increase in the runtime of the models. This fact justifies the proposal of the hybrid constructive heuristics to solve instances with a greater number of jobs. Hence, the upper bounds of the proposed H1 and H2 heuristics are compared with the best upper bound found by the complete model. The upper bound of the heuristics are

Table 4: Average computational results for complete model and constructive heuristics.

Group 1 - Processing time [1, 10]										Group 2 - Processing time [10, 100]									
n	m		MIP		JB		H1		H2		MIP		JB		H1		H2		
			GAP	T(s)	GAP	T(s)	GAP	T(s)	GAP	T(s)	GAP	T(s)	GAP	T(s)	GAP	T(s)	GAP	T(s)	
5	3	Best	0.0%	0.1	0.0%	0.0	0.0%	0.0	0.0%	0.0	0.0%	6.7	0.0%	0.0	0.0%	0.0	0.0%	0.0	
		Average	0.0%	0.2	6.8%	0.0	4.8%	0.0	4.8%	0.0	0.0%	21.0	2.4%	0.0	2.3%	0.0	2.3%	0.0	
	4	Best	0.0%	0.1	0.0%	0.0	0.0%	0.0	0.0%	0.0	0.0%	3.9	0.0%	0.0	0.0%	0.0	0.0%	0.0	
		Average	0.0%	0.3	7.7%	0.0	5.9%	0.0	5.9%	0.0	0.0%	354.9	7.5%	0.0	6.2%	0.0	6.2%	0.0	
	5	Best	0.0%	0.3	4.5%	0.0	0.0%	0.0	0.0%	0.0	0.0%	32.6	3.8%	0.0	0.0%	0.0	0.0%	0.0	
		Average	0.0%	0.5	9.8%	0.0	2.5%	0.0	3.3%	0.0	0.8%	1570.1	9.9%	0.0	2.6%	0.0	2.6%	0.0	
	6	Best	0.0%	0.1	4.8%	0.0	0.0%	0.0	0.0%	0.0	0.0%	80.9	5.7%	0.0	0.0%	0.0	0.0%	0.0	
		Average	0.0%	1.3	16.6%	0.0	5.2%	0.0	5.5%	0.0	11.8%	2388.3	16.1%	0.0	4.8%	0.0	6.2%	0.0	
	7	Best	0.0%	0.2	0.0%	0.0	0.0%	0.0	0.0%	0.0	0.0%	94.9	0.0%	0.0	0.0%	0.0	0.0%	0.0	
		Average	0.0%	3.7	14.1%	0.0	6.3%	0.0	5.4%	0.0	13.1%	2882.5	13.9%	0.0	5.1%	0.0	6.7%	0.0	
	Subgroup Average			0.0%	1.2	11.0%	0.0	4.9%	0.0	5.0%	0.0	5.1%	1443.4	10.0%	0.0	4.2%	0.0	4.8%	0.0
10	6	Best	0.0%	5.1	0.0%	0.0	0.0%	0.0	0.0%	0.0	28.7%	2801.1	0.7%	0.0	0.0%	0.0	0.0%	0.0	
		Average	0.0%	20.2	11.8%	0.0	1.8%	0.0	1.7%	0.0	28.2%	3509.9	11.3%	0.0	1.4%	0.0	1.4%	0.0	
	8	Best	0.0%	10.2	2.7%	0.0	0.0%	0.0	0.0%	0.0	23.6%	3597.6	2.3%	0.0	0.0%	0.0	0.0%	0.0	
		Average	0.0%	167.5	13.3%	0.0	6.0%	0.0	5.4%	0.0	36.7%	3600.0	17.0%	0.0	3.7%	0.0	4.2%	0.0	
	10	Best	0.0%	14.1	1.4%	0.0	0.0%	0.0	2.2%	0.0	35.7%	3600.0	2.2%	0.0	4.6%	0.0	1.0%	0.0	
		Average	0.0%	369.6	15.9%	0.0	6.5%	0.0	6.4%	0.0	45.2%	3600.0	18.4%	0.0	7.6%	0.0	7.9%	0.0	
	12	Best	0.0%	42.1	3.9%	0.0	0.0%	0.0	0.0%	0.0	51.7%	3600.0	3.1%	0.0	0.0%	0.0	0.0%	0.0	
		Average	5.0%	2440.5	14.0%	0.0	5.4%	0.0	5.3%	0.0	65.2%	3600.0	13.9%	0.0	5.4%	0.0	5.4%	0.0	
	14	Best	0.0%	21.9	1.3%	0.0	0.0%	0.0	1.0%	0.0	53.2%	3600.0	2.0%	0.0	0.0%	0.0	0.0%	0.0	
		Average	4.6%	3141.6	10.8%	0.0	7.0%	0.0	8.1%	0.0	73.6%	3600.0	11.4%	0.0	6.7%	0.0	5.8%	0.0	
	Subgroup Average			1.9%	1227.9	13.1%	0.0	5.3%	0.0	5.4%	0.0	49.8%	3582.0	14.4%	0.0	5.0%	0.0	4.9%	0.0
20	12	Best	1.5%	3600.0	10.0%	0.0	1.5%	0.0	0.7%	0.0	-	3600.0	10.2%	0.0	0.6%	0.0	0.6%	0.0	
		Average	14.8%	3600.0	18.1%	0.0	6.6%	0.0	7.1%	0.0	-	3600.0	17.9%	0.0	6.4%	0.0	6.5%	0.0	
	16	Best	14.5%	3600.0	16.4%	0.0	1.6%	0.0	5.4%	0.0	-	3600.0	17.4%	0.0	2.3%	0.0	4.2%	0.0	
		Average	32.4%	3600.0	23.0%	0.0	9.8%	0.0	10.5%	0.0	-	3600.0	23.3%	0.0	8.2%	0.0	10.4%	0.0	
	20	Best	24.5%	3600.0	14.3%	0.0	2.5%	0.0	4.4%	0.0	-	3600.0	15.5%	0.0	3.7%	0.0	1.3%	0.0	
		Average	36.0%	3600.0	24.2%	0.0	11.1%	0.0	10.9%	0.0	-	3600.0	24.8%	0.0	11.4%	0.0	10.3%	0.0	
	24	Best	19.9%	3600.0	2.4%	0.0	0.0%	0.0	2.6%	0.0	-	3600.0	1.5%	0.0	0.6%	0.0	5.2%	0.0	
		Average	38.4%	3600.0	21.2%	0.0	12.8%	0.0	14.0%	0.0	-	3600.0	21.7%	0.0	12.2%	0.0	13.0%	0.0	
	28	Best	19.4%	3600.0	1.6%	0.0	1.1%	0.0	4.9%	0.0	-	3600.0	3.1%	0.0	0.9%	0.0	0.9%	0.0	
		Average	36.5%	3600.0	16.7%	0.0	15.2%	0.0	15.0%	0.0	-	3600.0	17.4%	0.0	14.4%	0.0	13.5%	0.0	
	Subgroup Average			31.6%	3600.0	20.6%	0.0	11.1%	0.0	11.5%	0.0	*	3600.0	21.0%	0.0	10.5%	0.0	10.7%	0.0
40	24	Best	40.2%	3600.0	15.7%	0.0	3.9%	0.0	4.5%	0.0	-	3600.0	2.3%	0.0	0.0%	0.0	1.8%	0.0	
		Average	46.3%	3600.0	20.7%	0.0	9.6%	0.0	9.0%	0.0	-	3600.0	8.6%	0.0	6.2%	0.0	5.6%	0.0	
	32	Best	52.6%	3600.0	21.3%	0.0	9.0%	0.0	8.3%	0.0	-	3600.0	22.4%	0.0	6.0%	0.0	6.0%	0.0	
		Average	57.0%	3600.0	26.5%	0.0	12.8%	0.0	12.5%	0.0	-	3600.0	28.5%	0.0	11.8%	0.0	11.0%	0.0	
	40	Best	57.3%	3600.0	26.1%	0.0	9.1%	0.0	8.3%	0.0	-	3600.0	24.4%	0.0	9.0%	0.0	10.4%	0.0	
		Average	69.2%	3600.0	31.1%	0.0	12.5%	0.0	12.3%	0.0	-	3600.0	32.4%	0.0	13.3%	0.0	13.6%	0.0	
	48	Best	-	3600.0	23.9%	0.0	9.5%	0.0	11.6%	0.0	-	3600.0	23.0%	0.0	10.8%	0.0	11.6%	0.0	
		Average	-	3600.0	29.6%	0.0	16.6%	0.0	16.8%	0.0	-	3600.0	29.1%	0.0	16.6%	0.0	16.1%	0.0	
	56	Best	-	3600.0	16.8%	0.0	13.2%	0.0	15.5%	0.0	-	3600.0	18.4%	0.0	13.8%	0.0	14.8%	0.0	
		Average	-	3600.0	23.3%	0.0	19.7%	0.0	19.3%	0.0	-	3600.0	23.3%	0.0	18.7%	0.0	18.1%	0.0	
	Subgroup Average			*	3600.0	26.2%	0.0	14.2%	0.0	14.0%	0.0	*	3600.0	24.4%	0.0	13.3%	0.0	12.9%	0.0
60	36	Best	-	3600.0	16.9%	0.0	5.5%	0.0	4.5%	0.0	-	3600.0	17.8%	0.0	5.9%	0.0	3.4%	0.0	
		Average	-	3600.0	22.8%	0.0	8.6%	0.0	8.4%	0.0	-	3600.0	22.7%	0.0	8.4%	0.0	7.6%	0.0	
	48	Best	-	3600.0	26.0%	0.0	9.6%	0.0	10.7%	0.0	-	3600.0	26.5%	0.0	9.4%	0.0	8.5%	0.0	
		Average	-	3600.0	29.9%	0.0	12.0%	0.0	12.8%	0.0	-	3600.0	29.6%	0.0	11.2%	0.0	12.0%	0.0	
	60	Best	-	3600.0	26.6%	0.0	14.2%	0.0	13.9%	0.0	-	3600.0	28.0%	0.0	13.3%	0.0	12.9%	0.0	
		Average	-	3600.0	33.5%	0.0	18.2%	0.0	18.3%	0.0	-	3600.0	33.6%	0.0	16.9%	0.0	17.0%	0.0	
	72	Best	-	3600.0	25.7%	0.0	15.5%	0.0	14.8%	0.0	-	3600.0	26.0%	0.0	13.5%	0.0	13.5%	0.0	
		Average	-	3600.0	30.6%	0.0	19.1%	0.0	18.9%	0.0	-	3600.0	30.5%	0.0	18.2%	0.0	17.6%	0.0	
	84	Best	-	3600.0	20.3%	0.0	19.3%	0.0	18.4%	0.0	-	3600.0	20.2%	0.0	17.8%	0.0	17.4%	0.0	
		Average	-	3600.0	26.0%	0.0	21.9%	0.0	21.4%	0.0	-	3600.0	25.9%	0.0	21.4%	0.0	20.5%	0.0	
	Subgroup Average			*	3600.0	28.6%	0.0	16.0%	0.0	16.0%	0.0	*	3600.0	28.5%	0.0	15.2%	0.0	14.8%	0.0

better than those found by the complete model in almost all instances tested (92% of all tested instances), only in four instances the model presented a better outcome.

Comparing the GAP values of the MIP, LR and HgR we can observe that when considering processing times between 1 and 10 (group 1) the MIP has low GAP values for $n = 5$ jobs, with subgroup average GAP= 0%, while the LR has subgroup average GAP= 34.5% and HgR reaches a subgroup average GAP equals 4.8%. For $n = 10$ jobs, the MIP still has a low subgroup average GAP (1.9%) whereas the subgroup average GAP for the LR is 45.4% and for the HgR is 5.2%. When the number of jobs increases to $n = 20$ the subgroup average GAP of the MIP increases considerably (reaches 31.3%), for the LR is 57.7% while the HgR has a subgroup average GAP of 11.0%. For larger instances of group 1 it is not possible to calculate the value of subgroup average GAP for the MIP either the LR, since the methods do not find all values, while the HgR finds a subgroup average GAP equals 13.9% for $n = 40$ jobs and 16.0% for $n = 60$ jobs.

When considering processing times between 10 and 100 (group 2), the Hybrid Lagrangean Metaheuristic Framework subgroup average GAPs are noticeably better than the complete model and the linear relaxation. For $n = 5$ jobs the HgR has a subgroup average GAP= 4.2% while the MIP reaches 5.0% and for the LR is 35.3%. For $n = 10$ jobs the HgR get a GAP= 4.8% in average, and MIP and LR has an average GAP of 49.2% and 54.0%, respectively. For $n = 20$, $n = 40$ and $n = 60$ jobs it is not possible to calculate the value of subgroup average GAP for the MIP either the LR, while the HgR finds a subgroup average GAP equals 10.1% for $n = 20$, 12.8% for $n = 40$ and 14.8% for $n = 60$ jobs.

In Table 4 we report computational results for the MIP, JB, H1 and H2. The GAP is calculated considering the best lower bound found when comparing the lower bounds of Complete Model and Hybrid Lagrangean Metaheuristic Framework. It is worth highlighting that the constructive heuristic JB is proposed by Chen and Lee [20] and we aim to compare its strength with those (H1 and H2) proposed in this work.

The heuristic JB is proposed in two steps. Firstly, for an instance of $F2|CD|C_{max}$, the authors construct an instance of $F2||C_{max}$ with n jobs. The first stage is maintained unchangeable and the second stage is converted into n jobs. Secondly, Johnson's algorithm is applied in order to obtain a sequence in the first stage, generating a lower bound. From the sequence in the first stage, jobs $j \in J_2$ at second stage are sequenced as soon as possible in the time horizon, respecting the completion time of its predecessors. We run the above algorithm for the primary problem and its reverse problem and select the best solution. As a result, an Upper Bound for the problem is generated (See [20] for a more detailed explanation), so we calculated the relative GAP. In a similar way to the heuristics H1 and H2, the NEH algorithm and Local search is used to refine the results of JB.

In general, the hybrid heuristics had a similar average GAP, being around 10% for both groups, while the heuristic JB had an average GAP of about 20%. Based on the Table 4, the heuristic H2 showed GAP results more efficient than H1 and JB. Comparing the average GAPs we can observe that, for $n = m$, H2 gives the best performance in 60% of the instances for the first group and approximately 40% of the instances for the second group. H1 reaches best GAPs in 40% of the cases in group 1 and 80% of the cases in group 2. JB had no best GAP in any instance. In the same way, it is noticeable that when $n < m$ occurred, H2 is the best in 60% and 70% of the instances of groups 1 and 2 respectively, followed by H1, with 40% for both groups. When $n > m$, the heuristic H2 is the best option in 70% and 60% of the cases of group 1 and 2. H1 in 50% and 70% and JB had no best GAP.

Summarizing, the heuristic H2 has its better performance when $n < m$ for small and large instances. When $n > m$, the heuristic H2 gets better for small cases, while H1 presents better results for large instances. The same behavior is noticed when $n = m$, H2 is best for small cases, and H1 for larger ones.

6. Conclusions

In this work, we develop efficient ways to solve the two-machine cross-docking problem. We analyze a time-indexed formulation and a Hybrid Lagrangean Metaheuristic Framework. As expected, the performance of the MIP is strongly dependent on the instances size, being not able of solving the problem with medium and large dimensions.

It is worth noticing that, even having polynomially solvable Lagrangean sub-problems, the possibility of improving the makespan at each iteration, transforms the problem at each step, improving its linear relaxation bounds. The HgR shows an efficient performance, obtaining good bounds in a reduced computational time.

The heuristics proved to work very well with the Lagrangean approach. In a very effective way, H1 and H2 find goods upper bounds, outperforming previous results. The heuristic H2 gives the best average GAP results for 64% and 60% of the cases in group 1 and 2, whereas H1 achieved the best GAPs in 44% and 60%, of each respective group.

References

References

- [1] ALBA, E. Parallel metaheuristics: a new class of algorithms. *Wiley.com*. 47. (2005).
- [2] ALPAN, G., LARBI, R., AND PENZ, B. A bounded dynamic programming approach to schedule operations in a cross docking platform. *Computers & Industrial Engineering*. 60(3): (2011), 385–396.
- [3] ARABANI, A. B., GHOMI, S. F., AND ZANDIEH, M. Meta-heuristics implementation for scheduling of trucks in a cross-docking system with temporary storage. *Expert systems with Applications*. 38(3): (2011), 1964–1979.

- [4] ARAÚJO, D. P. M., AND MELO, B. M. R. Heurísticas construtivas para o sequenciamento de caminhões em centros de cross-docking. Master's thesis, Universidade Federal de Minas Gerais, 2010.
- [5] ARROYO, J., NUNES, G., AND KAMKE, E. Iterative local search heuristic for the single machine scheduling problem with sequence dependent setup times and due dates. in: Hybrid intelligent systems. *Ninth International Conference. 1:* (2009), 505–510.
- [6] BAHENSE, L., MACULAN, N., AND SAGASTIZABAL, C. The volume algorithm revised: relation with bundle methods. *Mathematical Programming* 94. (2002), 41–69.
- [7] BARAHONA, F., AND ANBIL, R. The volume algorithm: producing primal solutions with a subgradient method. *Mathematical Programming* 87(3) (2000), 385–399.
- [8] BARAHONA, F., AND LADÁNYI, L. Branch and cut based on the volume algorithm: Steiner trees in graphs and max-cut. *RAIRO - Operations Research* 40 (2006), 53–73.
- [9] BARTHOLDI, J., AND GUE, K. Reducing labor costs in an ltl crossdocking terminal. *Operations Research*. 48(6): (2002), 823–832.
- [10] BARTZ-BEIELSTEIN, T. An r package for automatic and interactive tuning of optimization algorithms by sequential parameter optimization. *arXiv preprint arXiv:1006.4645*. (2010.).
- [11] BARTZ-BEIELSTEIN, T., AND ZAEFFERER, M. A gentle introduction to sequential parameter optimization. *Technical Report 2, Bibliothek der Fachhochschule Koeln*. (2012.).
- [12] BELLE, J. V., VALCKENAERS, P., AND CATTRYSSSE, D. Cross-docking: State of the art. *Omega*. 40: (2012), 827–846.

- [13] BLUM, C., PUCHINGER, J., RAIDL, G., AND ROLI, A. Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing*. 11(6): (2011), 4135–4151.
- [14] BOSCHETTI, M., AND MANIEZZO, V. Benders decomposition, lagrangean relaxation and metaheuristic design. *Journal of Heuristics*. 15(3): (2009), 283–312.
- [15] BOYSEN, N. Truck scheduling at zero-inventory cross docking terminals. *Computers & Operations Research*. 37(1): (2010), 32–41.
- [16] BOYSEN, N., AND FLIEDNER, M. Cross dock scheduling: Classification, literature review and research agenda. *Omega*. 38(6): (2010), 413–422.
- [17] BOYSEN, N., FLIEDNER, M., AND SCHOLL, A. Scheduling inbound and outbound trucks at crossdocking terminals. *OR spectrum*. 32(1): (2010), 135–161.
- [18] BOZER, Y., AND CARLO, H. Optimizing inbound and outbound door assignments in less-than-truckload crossdocks. *IIE Transactions*. 40: (2008), 1007–1018.
- [19] CAMPBELL, J. A survey of network hub location. *Studies in Locational Analysis*. 6: (1994), 31–49.
- [20] CHEN, F., AND LEE, C.-Y. Minimizing the makespan in a two-machine cross-docking flow shop problem. *European Journal of Operational Research*. 193(1): (2009), 59–72.
- [21] CHEN, F., AND SONG, K. Minimizing makespan in two-stage hybrid cross docking scheduling problem. *Computers & Operations Research*. 36(6): (2009), 2066–2073.
- [22] CHEN, J. A hybrid heuristic for the uncapacited single allocation hub location problem. *Omega*. 35: (2007), 211–220.

- [23] CHEN, P., GUO, Y., LIM, A., AND RODRIGUES, B. Multiple crossdocks with inventory and time windows. *Computers and Operations Research*. 33: (2006), 43–46.
- [24] CLAUSEN, J., CORDEAU, J., LAPORTE, G., WEN, M., AND J., L. Vehicle routing scheduling with cross-docking. *Journal of the Operational Research Society*. 60: (2009), 1708–1718.
- [25] COTA, P. M., GIMENEZ, B. M., ARAÚJO, D. P., NOGUEIRA, T. H., DE SOUZA, M. C., AND RAVETTI, M. G. Time-indexed formulation and polynomial time heuristic for a multi-dock truck scheduling problem in a cross-docking centre. *Computers & Industrial Engineering*. 95: (2016), 135–143.
- [26] FONSECA, G. B. O problema de sequenciamento de caminhões em um centro de cross-docking com duas máquinas. Master’s thesis, Universidade Federal de Minas Gerais, 2015.
- [27] FORGER, G. Ups starts worlds premiere cross-docking operation. *Modern material handling*. 36(8): (1995), 36–38.
- [28] FUKUDA, E. H. Algoritmo de volume e otimização não diferenciável. Master’s thesis, USP, 2007.
- [29] GUE, K. R. The effects of trailer scheduling on the layout of freight terminals. *Transportation Science*. 33(4): (1999), 419–428.
- [30] KIM, C., YANG, K. H., AND KIM, J. A strategy for third-party logistics systems: a case analysis using the blue ocean strategy. *Omega*. 36(4): (2008), 522–534.
- [31] KLOSE, A., AND DREXL, A. Facility location models for distribution system design. *European Journal of Operational Research*. 162: (2005), 4–29.

- [32] LADIER, A-L. AND ALPAN, G. Cross-docking operations: Current research versus industry practice. *Omega*. 62: (2016), 145–162.
- [33] LARBI, R., ALPAN, G., BAPTISTE, P., AND PENZ, B. Scheduling cross docking operations under full, partial and no information on inbound arrivals. *Computers & Operations Research*. 38(6): (2011), 889–900.
- [34] LEE, K., LEE, Y., AND JUNG, J. Positioning of goods in a cross-docking environment. *Computers & Industrial Engineering*. 54(3): (2008), 677–689.
- [35] LEMARÉCHAL, C. An extension of davidon methods to non differentiable problems. in: Nondifferentiable optimization. *Springer* (1975), 95–109.
- [36] LEMARÉCHAL, C. Lagrangean relaxation, computational combinatorial. *Springer Verlag*. (2001).
- [37] LEMARÉCHAL, C. Nondifferentiable optimization. *Optimization, Handbooks in Operations Research* (529-572).
- [38] LIMA, M. F. O problema de sequenciamento de caminhões numa estação de cross-docking com duas máquinas: Formulação indexada no tempo, relaxação lagrangeana e geração de colunas. Master’s thesis, Universidade Federal de Minas Gerais, 2014.
- [39] LIRA, E. G. Um algoritmo iterated greedy para o problema de sequenciamento de caminhões em centros de cross-docking. Master’s thesis, Universidade Federal de Minas Gerais, 2013.
- [40] MCWILLIAMS, D. L. Iterative improvement to solve the parcel hub scheduling problem. *Computers & Operations Research*. 59(1): (2010), 136–144.
- [41] MIAO, Z., LIM, A., AND MA, H. Truck dock assignment problem with operational time constraint within crossdocks. *European journal of operational research*. 192(1): (2009), 105–115.

- [42] NAWAZ, M., ENSCORE, J., AND HAM, I. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega*. 11(1): (1983), 91–95.
- [43] NOGUEIRA, T. H. *Single machine scheduling problems with sequence-dependent setup times*. PhD thesis, Universidade Federal de Minas Gerais, 2014.
- [44] OH, Y., HWANG, H., CHAB, C., AND LEE, S. A dock-door assignment problem for the korean mail distribution center. *Computers & Industrial Engineering*. 51(2): (2006), 288–296.
- [45] PAULA, M., MATEUS, G., AND RAVETTI, M. A non-delayed relax-and-cut algorithm for scheduling problems with parallel machines, due dates and sequence-dependent setup times. *Computers & Operations Research*. 37(5): (2010), 938–949.
- [46] PINEDO, M. Scheduling: Theory, algorithms and systems. *Hall, Englewood Cliffs*. (2008.).
- [47] PIRKWIESER, S., RAIDL, G., AND PUCHINGER, J. Combining lagrangian decomposition with an evolutionary algorithm for the knapsack constrained maximum spanning tree problem. in: *Evolutionary computation in combinatorial optimization*. Springer. (2007), 176–187.
- [48] PUCHINGER, J., AND RAIDL, G. Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification. in: *Artificial intelligence and knowledge engineering applications: a bioinspired approach*. Springer (2005), 41–53.
- [49] SAVELSBERGH, M., AND WOENSEL, T. V. City logistics: Challenges and opportunities. *Transportation Science* 50, 2 (2016), 579–590.
- [50] SMITH, W. E. Varius optimizers for single-stage production. *Naval Res. Logist.* 3: (1956), 59–66.

- [51] STALK, G., EVANS, P., AND SHULMAN, L. E. Competing on capabilities: the new rules of corporate strategy. *Harvard business review*. 70(2): (1991), 57–69.
- [52] STUTZLE, T. *Local search algorithms for combinatorial problems*. PhD thesis, Darmstadt University of Technology, 1998.
- [53] TSUI, L., AND CHANG, C. A microcomputer based decision support tool for assigning dock doors in freight yards. *Computers & Industrial Engineering*. 19: (1990), 309–312.
- [54] TSUI, L., AND CHANG, C. An optimal solution to a dock door assignment problem. *Computers & Industrial Engineering*. 23: (1992), 283–286.
- [55] VAHDANI, B., AND ZANDIEH, M. Scheduling trucks in cross-docking systems: Robust metaheuristics. *Computers & Operations Research*. 58(1): (2010), 12–24.
- [56] WITT, C. E. Crossdocking: Concepts demand choice. *Material Handling Engineering*. 53(7): (1998), 44–49.
- [57] WOLFE, P. A method of conjugate subgradients for minimizing nondifferentiable functions. in: Nondifferentiable optimization. *Springer* (1975), 145–173.

Appendix A Considerations about WSPT-TRD

Theorem 2. *Considering the problem $1||C_{max} + \sum I_j W_j$, the algorithm WSPT-TRD obtains an optimal solution.*

Proof 2. *The objective function has two criteria, C_{max} and $\sum I_j W_j$. Regardless of the situation, C_{max} , will always aim to allocate the jobs as soon as possible. However, the $\sum I_j W_j$ depend of the weights values. Therefore, the proof is divided into two parts.*

Part I: If $w_j^2 \geq 0$ (see Figure 3).

In this case, the two criteria of the objective function are not in conflict. The criteria C_{max} and $\sum I_j W_j$ aims to schedule the jobs as soon as possible. As there is no idle time between jobs, for C_{max} any sequence starting at beginning of time horizon (t_i) is optimal. In this way, the only existing criteria becomes $\sum I_j W_j$ and its optimal WSPT rule is optimal for the problem.

Part II: If $w_j^2 < 0$.

In this case, the criteria are in conflict. C_{max} aim to schedule the jobs at beginning (t_i) of the time horizon, while $\sum I_j W_j$ at the end (t_f). Consider an optimal sequence generated by WSPT rule with one subset S' of jobs with $\sum w_j^2 > -1$. Suppose, by contradiction, that this subset must be allocated at the end (t_f) of the time horizon. Initially the sequence is all allocated at the beginning (t_i) of the time (see Figure 4a). When the schedule S' with $w_j^2 < 0$ is moved in one unit of the time horizon, its C_{max} increase one unit, but $\sum I_j W_j$ decrease $\sum w_j^2$. As $C_{max} - \sum w_j^2 > 0$ the objective function increase and the optimality is contradicted. Therefore, the $\sum w_j^2$ must be less or equal to 1 for allocate the jobs at the end (see Figure 4b). It should be noted that for the case $\sum w_j^2 = -1$, these jobs are indifferent. \square

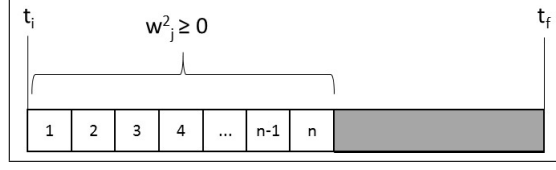
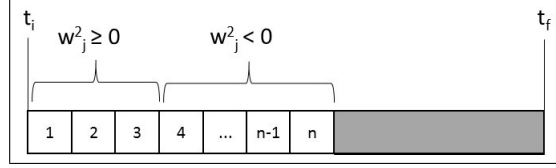


Figure 3: Sequence with positive weights



(a) Initial Schedule



(b) Optimal Schedule

Figure 4: Sequence with negative and positive weights

Appendix B Sequential Parameter Optimization Toolbox (SPOT)

Due to its high efficiency, the Sequential Parameter Optimization Toolbox (SPOT) is the defined method to determine the parameters vector of Volume Algorithm. A parameters vector comprises the values of different parameters of the Volume Algorithm, i.e. it represents a candidate solution of a parameter setting. SPOT is an implementation of the Sequential Parameter Optimization (SPO), which is an iterative model-based method of tuning algorithms. The tuning process is based on the parameters data, and their utility delivered by the performance of the volume algorithm. SPO performs a multi-stage procedure where the model is updated at each iteration with a set of new vectors and new predictions of utility in order to improve the algorithm's efficiency.

The goal of SPOT is the determination of good parameters settings for heuristic algorithms. It provides statistical tools for analyzing and understand-

ing algorithm’s performance. SPOT is implemented as a R package, and is available in the R archive network at <http://cran.r-project.org/web/packages/SPOT/index.html>. Further explanation about SPOT can be obtained from Bartz-Beielstein [10], which provides an exemplification on how SPOT can be used for automatic and iterative tuning. Bartz-Beielstein and Zaefferer [11] also give an introductory overview about tuning with SPOT.

The key elements of the SPOT methodology are algorithm design (D_a) and problem design (D_p). The first one defines ranges of parameter values that influence the behaviour of an algorithm, such as crossover rate. These parameters are treated as variables $p_a \in D_a$ in the tuning algorithm, where p_a represents the vector of parameters settings. D_p refers to variables related to the tuning optimization problem, e.g. the search space dimension.

SPOT is composed by two phases: the build of the model and its sequential improvement. Phase 1 determinates an initial designs population from the algorithm’s parameter space. The observed algorithm is run k times for each design, where k is the number of repetitions performed for each parameter setting and is increased in each run. Phase 2 leads to the efficiency of the approach and is characterized by the following loop:

- update the model given the obtained data;
- generate design points and predict their utility by sampling the model;
- choose the best design vectors and run $k + 1$ times the algorithm for each of them;
- new design points are added to the population and the loop restarts if the termination criteria is not reached.

The loop simulates the use of different parameters settings, it is subject to interactions between parameters and random effects into the experiment. SPOT uses results from algorithm runs to build up a meta model to tune algorithms in a reproducible way.

B.1 Experimental setup

Classical works about SPOT define three different layers to analyse parameter tuning. The first one is the *application layer*, considered in our case as the two-machine flow shop problem with cross-docking constraints. The objective function and the problem parameters are defined at this layer. The *algorithm layer*, is related to the representation of the heuristic algorithm and its required parameters are the ones that determine the algorithm’s performance. And finally, the *design layer*, where is the tuning method, tries to find good parameter settings for the algorithm layer.

In this way we face two optimization problems: problem solving and parameter tuning. The problem solving covers the application layer and the Volume Algorithm of the algorithm layer and aims to find an optimal solution for the problem. The parameters tuning uses a tuning method to find the best parameter values for the Volume Algorithm based on lower bounds found. The fitness value is the quality measure of the first optimization, which depends on the problem instance to be solved. Utility is the quality measure for parameter tuning, which reflects the performance of the Volume Algorithm for a vector of parameters.

Thereby, the experimental setup consists of an application layer represented by twenty instances of the two-machine flow shop problem, showed on Table 5. To define the region of interest (ROI) of the tuning algorithm, the type and the lower and upper bounds of the volume algorithm’s parameters are summarized on Table 6.

Table 5: Sampling of instances

Type	Instances
Y1	5-3-4-1
Y2	5-4-4-1
Y3	5-5-4-1
Y4	5-6-4-1
Y5	5-7-4-1
Y6	10-6-9-1
Y7	10-8-9-1
Y8	10-10-9-1
Y9	10-12-9-1
Y10	10-14-9-1
Y11	5-3-4-2
Y12	5-4-4-2
Y13	5-5-4-2
Y14	5-6-4-2
Y15	5-7-4-2
Y16	10-6-9-2
Y17	10-8-9-2
Y18	10-10-9-2
Y19	10-12-9-2
Y20	10-14-9-2

B.2 Results of the experiment

The SPOT algorithm, implemented in R package, is connected with the volume algorithm implemented on C++ with the aid of one callString, as presented bellow. The computer used in the tests is a Intel (R) Xeon (R) CPU X5690 @ 3.47GHz with 24 processors, 132 GB of RAM, and Ubuntu Linux operating system.

Table 6: Parameter bounds for tuning the Volume Algorithm

Parameter	Lower bound	Upper bound	Type
π	0.0005	0.0100	FLOAT
MaxWaste	5	30	INT
factor	0.1	1.0	FLOAT
α_{max}	0.20	0.95	FLOAT
st	0.6	1.8	FLOAT
α	0.05	0.90	FLOAT
yellow	0.20	0.95	FLOAT
green	1.0	2.0	FLOAT

```

callString  $\leftarrow$  paste("./HgR ",  $\pi$ , MaxWaste, factor,  $\alpha_{max}$ , st,  $\alpha$ , yellow, green)
call  $\leftarrow$  system(callString, intern = TRUE)
#read the results
y = read.table("Results.txt")

```

The Table 7 presents the four best results found by SPOT method for the volume algorithm. The first column indicates the iteration of SPOT, the second one presents the indicated parameters vector and the third column indicate the solution value obtained by the tested instances (lower bounds).

The best parameters vector indicated for the volume algorithm, that generated the best lower bounds is $\pi=0.00679$, MaxWaste=24, factor=0.87753, $\alpha_{max}=0.30337$, st=1.41179, $\alpha=0.08300$, yellow=0.48159 and green=1.56487.

Table 7: Four best parameters settings for the volume algorithm

Iteration	Parameter								Solution value of the instances																			
	π	MaxWaste	factor	α_{max}	st	α	yellow	green	Y1	Y2	Y3	Y4	Y5	Y6	Y7	Y8	Y9	Y10	Y11	Y12	Y13	Y14	Y15	Y16	Y17	Y18	Y19	Y20
501	0.00708	28	0.22871	0.33265	1.01875	0.21709	0.80291	1.13994	24	14	30	20	30	36	46	56	60	66	155	137	191	328	414	337	537	558	629	793
502	0.00527	9	0.34752	0.75030	1.71397	0.73405	0.58396	1.56980	24	15	29	21	30	37	46	56	60	66	157	154	191	341	414	337	563	558	629	793
503	0.00688	20	0.62855	0.83046	1.17734	0.28130	0.60572	1.40758	25	15	30	21	30	37	46	56	60	66	118	136	191	307	414	337	531	558	629	793
504	0.00679	24	0.87753	0.30337	1.41179	0.08300	0.48159	1.56487	26	15	30	20	30	32	46	56	60	66	95	136	191	307	414	337	531	558	629	793